Assignment-5

```
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;
int xl=50,xh=200,yl=50,yh=200;
int flag=0;
float u1,v1,u2,v2;
struct code
{
  int t,b,r,l;
};
void init()
{
glClearColor(1,1,1,0);
 glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0,0,0);
}
code get_code(int u,int v)
{
        code c={0,0,0,0};
        if(u<xl)
        c.l=1;
        if(u>xh)
        c.r=1;
        if(v<yl)
        c.b=1;
        if(v>yh)
        c.t=1;
        return c;
}
```

```
/*
//#BRESENHEM LINE DRAWING ALOGORITHM
void line(int u1,int v1,int u2,int v2)
{
 int dx,dy,p,xi=1,yi=1;
 dx=u2-u1;
 dy=v2-v1;
 if(dx<0)
 {
        dx=-dx;
        xi=-1;
 }

 if(dy<0)
 {
        dy=-dy;
        yi=-1;
 }
 glBegin(GL_POINTS);
 glVertex2i(u1,v1);
 if(dx>dy)
 {
 p=(2*dy)-dx;
   while(u1!=u2)
   {
        if(p<=0)
        {
         p+=2*dy;
        }
        else
        {
         p+=2*(dy-dx);
```

```
            v1+=yi;
            }
            u1+=xi;
            glVertex2i(u1,v1);
    }
  }
  else
  {
    p=(2*dx)-dy;
     while(v1!=v2)
    {
            if(p<=0)
            {
             p+=2*dx;
            }
            else
            {
             p+=2*(dx-dy);
             u1+=xi;
            }
            v1+=yi;
            glVertex2i(u1,v1);
    }
  }
    glEnd();
  glFlush();
}
*/
void line(float u1,float v1,float u2,float v2)
{
 float dx,dy,x=u1,y=v1,xi,yi;
 int steps,i;
```

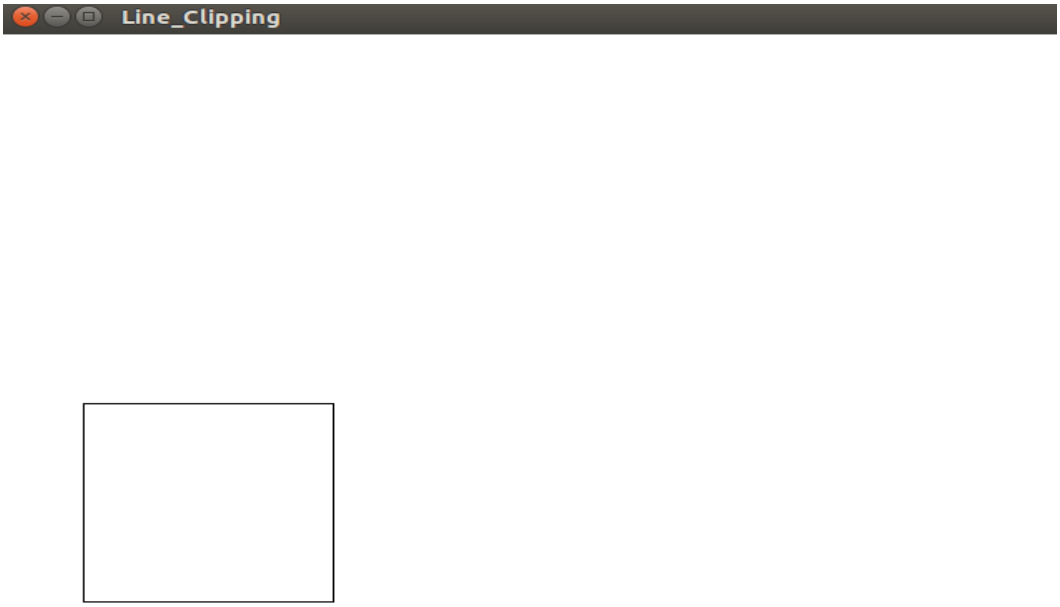
```
 dx=u2-u1;
 dy=v2-v1;
steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
 xi=dx/(float)steps;
 yi=dy/(float)steps;
 glBegin(GL_POINTS);
 glVertex2f(x,y);
 for(i=0;i<steps;i++)
 {
        x+=xi;
        y+=yi;
        glVertex2f(x,y);
 }
glEnd();
 glFlush();
}
void draw_window()
{
        line(50,50,200,50);
        line(50,50,50,200);
        line(200,50,200,200);
        line(50,200,200,200);
}
void mymouse(int button,int state,int x,int y)
{
        glColor3f(0,0,0);
        if(state==GLUT_DOWN && flag==0)
        {
                u1=x;
                v1=480-y;
                flag=1;
        }
```

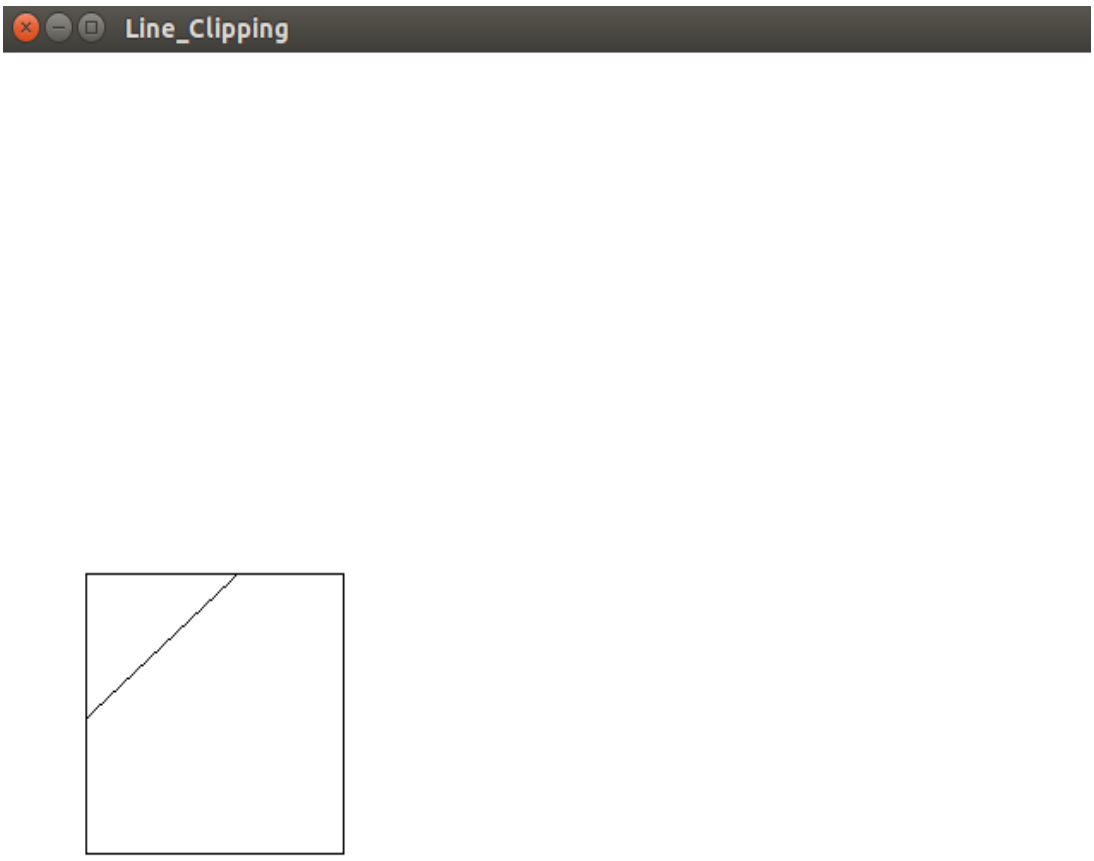
```
        else if(state==GLUT_DOWN && flag==1)
        {
                u2=x;
                v2=480-y;
                flag=2;
                line(u1,v1,u2,v2);
        }
}
void cohen()
{
        code c1,c2,c;
        float m;
        int xi,yi,flag;
        m=(v2-v1)/(u2-u1);
        c1=get_code(u1,v1);
        c2=get_code(u2,v2);
        while(1)
        {
                if( c1.t==0 && c2.t==0 && c1.b==0 && c2.b==0 && c1.r==0 && c2.r==0 && c1.l==0
&& c2.l==0 )
                break;
                else if( ( (c1.t  && c2.t) || (c1.b && c2.b) || (c1.r && c2.r) || (c1.l && c2.l) ) !=0)
                {
                        u1=v1=u2=v2=0;
                        break;
                }
                else
                {
                        if( c1.l==1 || c2.l==1)
                        {
                                xi=xl;
                                yi=v1+m*(xl-u1);
```

```
            if(c1.l==1)
            flag=0;

            else
            flag=1;

        }

        else if( c1.r==1 || c2.r==1 )
        {
            xi=xh;
            yi=v1+m*(xh-u1);

            if(c1.r==1)
            flag=0;

            else
            flag=1;
        }

        else if( c1.b==1 || c2.b==1 )
        {
            xi=u1+((1/m)*(yl-v1));
            yi=yl;

            if(c1.b==1)
            flag=0;

            else
            flag=1;
        }
        else if( c1.t==1 || c2.t==1 )
```

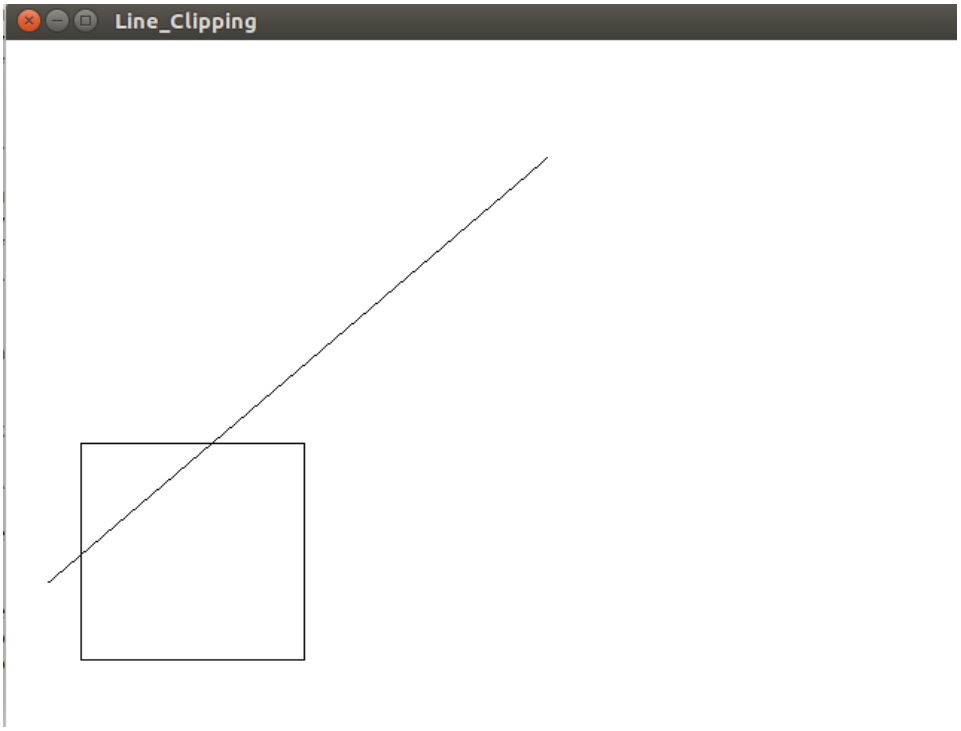
```
                        {
                                xi=u1+((1/m)*(yh-v1));
                                yi=yh;

                                if(c1.t==1)
                                flag=0;

                                else
                                flag=1;
                        }

                        c=get_code(xi,yi);

                        if(flag==0)
                        {
                                u1=xi;
                                v1=yi;
                                c1=c;
                        }
                        else if(flag==1)
                        {
                                u2=xi;
                                v2=yi;
                                c2=c;
                        }
                }//end_else
        }//end_while
        draw_window();
        line(u1,v1,u2,v2);
}
void mykey(char unsigned key,int x,int y)
{
```

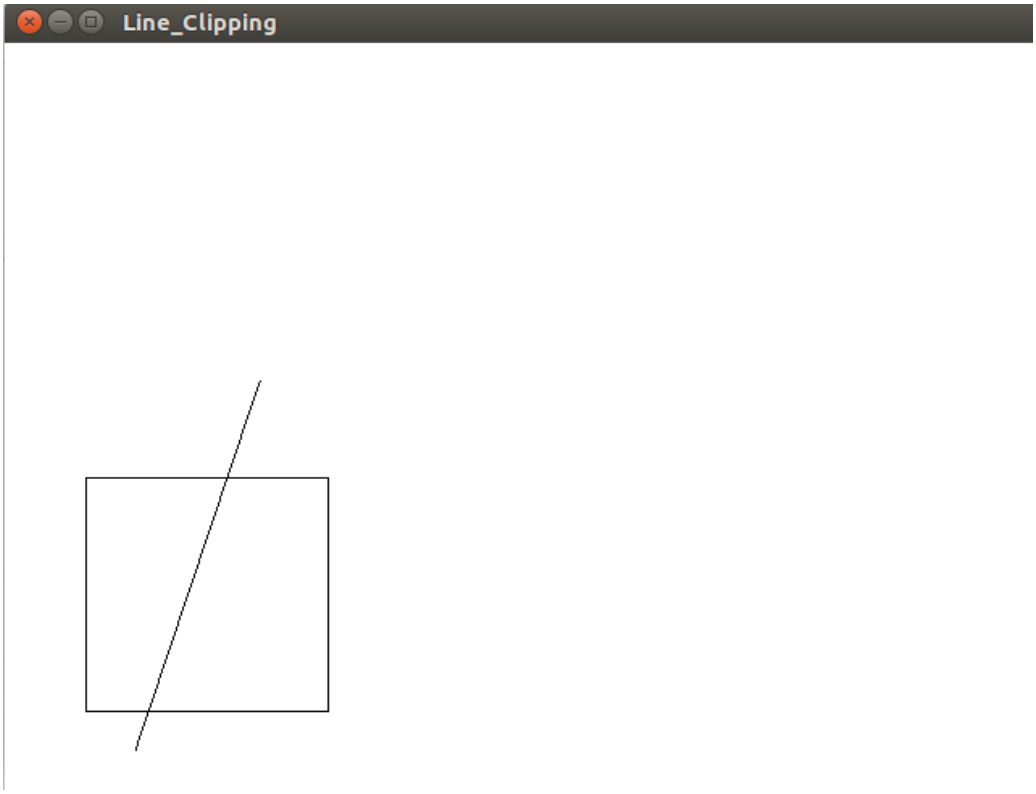
```
        if(key=='c')
        {
          init();
          cohen();
        }
        if(key=='r')
        {
          init();
          draw_window();
          flag=0;
        }
}
int main(int argc,char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE);
  glutInitWindowSize(640,480);
  glutInitWindowPosition(0,0);
  glutCreateWindow("Line_Clipping");
  gluOrtho2D(0,640,0,480);
  init();
  glFlush();
  draw_window();
  glutMouseFunc(mymouse);
  glutKeyboardFunc(mykey);
  glutMainLoop();
  return 0;
}
```


Line_Clipping


Line_Clipping


Line_Clipping


Line_Clipping


Line_Clipping


Line_Clipping

```
/*
Write a program in OpenGL on Linux Platform for clipping a line using Cohen Sutherland Outcode Method
*/
#include<stdio.h>                    //initial inclusions
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<math.h>
float xd1,yd1,xd2,yd2; //storing values for end points of line
int ymax=100; //initializing window coordinates
int ymin=-100;
int xmax=100;
int xmin=-100;
static int p=0;
void disp(); //declaring display function
float round_value(float v) //function to round value to next greater float
{
return (v+0.5);
}
void plotpoint(float a,float b)
{
glBegin(GL_POINTS);
glVertex2f(a,b);
glEnd();
}
void dda(float X1,float Y1,float X2,float Y2) //dda algorithm
{
float dx,dy,x,y,xinc,yinc;//initializations
int k,steps;
dx=X2-X1;                                    //difference of x coordinates
```

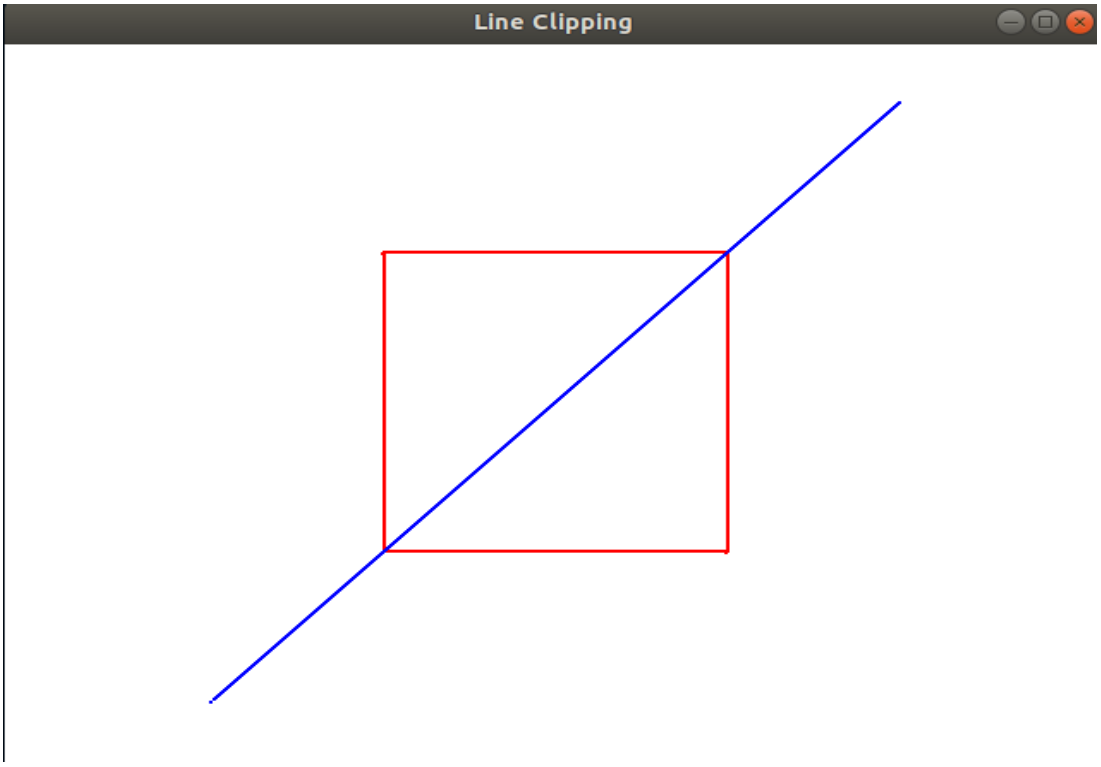
```
dy=Y2-Y1;                                                    //difference of y coordinates
steps=abs(dx)>abs(dy)?abs(dx):abs(dy); //calculation of number of steps
xinc=dx/(float)steps; //value for incrementing x
yinc=dy/(float)steps; //value for incrementing y
x=X1,y=Y1;
plotpoint(x,y);                                  //function to plot point on window
for(k=0;k<steps;k++) //loop to plot points
{
x+=xinc;                                                     //incrementing x by xinc
y+=yinc;                                                     //incrementing y by yinc
plotpoint(round_value(x),round_value(y)); //plotting point
}
glFlush();
}
int code(int x,int y) //calculating outcode of point
{
int c=0;
if(y>ymax) c=8;  //if greater than ymax set code to 8
if(y<ymin) c=4;                                   //if less than ymin set code to 4
if(x>xmax) c=c|2;                          //if greater than xmax set code to 2
if(x<xmin) c=c|1;                          //if less than ymin set code to 1
return c;
}
void cohen(float x1,float y1,float x2,float y2)      //implementing cohen-sutherland algorithm
{
int c1=code(x1,y1);                                          //checking for outcode of point 1
int c2=code(x2,y2);                                                  //checking for outcode of point 2
float m=(y2-y1)/(x2-x1);                   //checking slope of line
while((c1|c2)>0)                                                     //iterating loop till c1|c2>0
{
if((c1 & c2)>0)                                              //if both lie completely outside the window
```

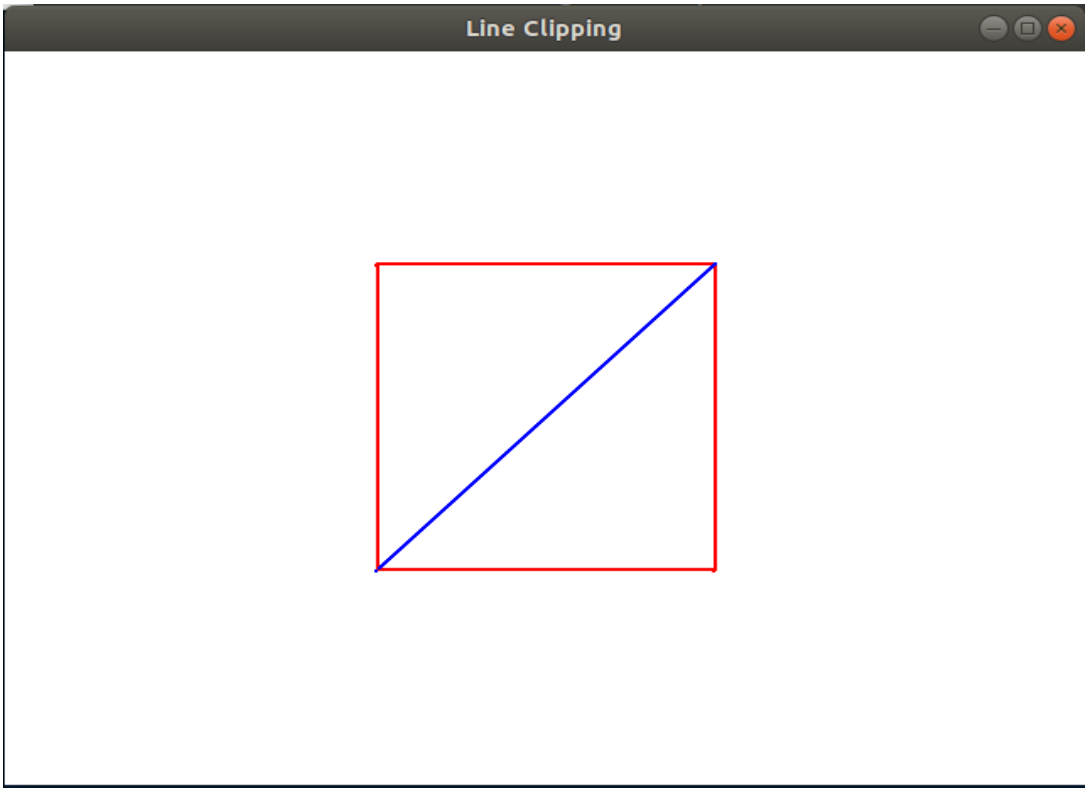
```
{
disp();
return;
}
int c;
float xi=x1;
float yi=y1;
c=c1;
float x,y;
if(c==0)                                        //checking if outcode is
equal to 0
{
c=c2;                                           //assigning outcode of c2
xi=x2;                                          //assigning x coordinate of
c2
yi=y2;                                          //assigning y coordinate of
c2
}
if((c & 8)>0)                                   //checking if  c&8 >0 ( greater than
ymax)
{
y=ymax;                                         //assigning new values to x
and y
x=xi+1.0/(m*(ymax-yi));
}
if((c & 4)>0)                                   //checking if c> 4 >0 (less than ymin)
{
y=ymin;                                         //assigning new values to x
and y
x=xi+1.0/(m*(ymin-yi));
}
if((c & 2)>0)                                   //checking if  c&2 >0 ( greater than
xmax)
```

```
{
x=xmax;
y=yi+m*(xmax-xi);
}
if((c & 1)>0)                                   //checking if  c&1 >0 (less than xmin)
{
x=xmin;
y=yi+m*(xmin-xi);
}
if(c==c1)                                       //checking code and
assigning new values
{
xd1=x;
yd1=y;
c1=code(xd1,yd1);
}
if(c==c2)                                       //checking code and
assigning new values
{
xd2=x;
yd2=y;
c2=code(xd2,yd2);
}
}
p++;
disp();                                         //calling
display function again to display new line
}
void mykey(unsigned char ch,int x,int y)
{
if(ch=='c')
{
```

```
cohen(xd1,yd1,xd2,yd2);                    //if character c is pressed calling algorithm
glFlush();


}
}
void disp()
{
glClear(GL_COLOR_BUFFER_BIT);//clearing buffer
glColor3f(1.0,0.0,0.0);                          //assigning color
dda(xmin,ymin,xmax,ymin);                  //creating window using dda algorithm to draw lines
dda(xmax,ymin,xmax,ymax);
dda(xmax,ymax,xmin,ymax);
dda(xmin,ymax,xmin,ymin);

glColor3f(0.0,0.0,1.0);                          //assigning color for line
dda(xd1,yd1,xd2,yd2);                          //drawing line
glFlush();
/*glBegin(GL_LINE_LOOP);
glVertex2i(-100,-100);
glVertex2i(100,-100);
glVertex2i(100,100);
glVertex2i(-100,100);
glEnd();
glColor3f(0.0,0.0,1.0);
glBegin(GL_LINES);
glVertex2i(xd1,yd1);
glVertex2i(xd2,yd2);
glEnd();*/
}
void init()
{
```

```
glClearColor(1.0,1.0,1.0,0);                //clearing background color to new color
glClear(GL_COLOR_BUFFER_BIT);               //clearing buffer
glPointSize(2);                                                    //assigning point size
gluOrtho2D(-320,320,-240,240);
glFlush();
}
int main(int argc,char **argv)
{
printf("Window coordinates are (-100,100,-100,100)\n");
printf("\nEnter coordinates of the line(limits : -320,320,-240,240) \nAfter entering enter c to clip\n");
printf("\nCoordinates of first point");
printf("\nX1: ");
scanf("%f",&xd1);                                                  //accepting value of x1
printf("\nY1: ");                                                  //accepting value of y1
scanf("%f",&yd1);
printf("\nCoordinates of second point");
printf("\nX2: ");
scanf("%f",&xd2);                                                  //accepting value of x2
printf("\nY2: ");                                                  //accepting value of y2
scanf("%f",&yd2);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowPosition(100,100);
glutInitWindowSize(640,480);
glutCreateWindow("Line Clipping");
init();
glutDisplayFunc(disp);
glutKeyboardFunc(mykey);
glutMainLoop();
return 0;
}
```


Line Clipping


Line Clipping

```
/** Sutherland Hodgeman Polygon Clipping

* Usage  - click on the canvas to seed the points of the polygon
*        press d - to draw the seeded polygon
*        drag the mouse cursor to draw the required clip rectangle
*        press c - to clip the required polygon section
* NOTE   - Clip rectangle must be drawn from top left to bottom right
*SContains Debug Code to reconstruct the clipping in comments in SHPC function
*/
#include <iostream>
#include <math.h>
#include <time.h>
#include <GL/glut.h>
#include <list>
using namespace std;
void init(){
glClearColor(1.0,1.0,1.0,1.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0,640,0,480);
}
int xmin = 0,ymin = 0,xmax = 0,ymax = 0;
int enter = 1,sz,st_flag=1;
float** pts;
class points{
int x;
int y;
public:
points(int x,int y){
this->x = x;
this->y = y;
```

```
}
int getx(){
return x;
}
int gety(){
return y;
}
};
class tryo{
int x;
int y;
public:
void setx(int x){this->x = x;}
int getx(){return x;}
};
points *s, *p;
list <points*> in;
list <points*> outer;
void delay(float ms){
clock_t goal = ms + clock();
while(goal>clock());
}
void drawPolygon(){
glBegin(GL_LINE_LOOP);
pts = new float*[in.size()];
for(int i=0; i<in.size(); i++){
pts[i] = new float[2];
}
sz = in.size();
while(in.size()>0){
points* temp = in.front();
```

```
pts[in.size()-1][0] = temp->getx();
pts[in.size()-1][1] = temp->gety();
glVertex2i(temp->getx(),temp->gety());
in.pop_front();
}
glEnd();
glFlush();
}
void redraw(){
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_LINE_LOOP);
for(int i=0; i<sz; i++){
glVertex2i(pts[i][0],pts[i][1]);
}
glEnd();
glFlush();
glColor3f(0,0,0);
glBegin(GL_LINE_LOOP);
glVertex2i(xmin,ymin);
glVertex2i(xmin,ymax);
glVertex2i(xmax,ymax);
glVertex2i(xmax,ymin);
glEnd();
glFlush();
glColor3f(1,0,0);
glLineWidth(1.0);
}
void draw_pixel(int x,int y)
{
glColor3f(1.0,0.0,0.0);
glPointSize(6.0);
```

```
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
}
int inside(int x, int y, int clip_edge){
switch(clip_edge){
case 1: if(x<xmax) return 1; else return 0;break;
case 2: if(y>ymax) return 1; else return 0;break;
case 3: if(x>xmin) return 1; else return 0;break;
case 4: if(y<ymin) return 1; else return 0;break;
default: return 0;break;
}
}
points* intersect(points* S, points* P, int clip_edge){
float m; //div by zero error earlier
if((P->getx()-S->getx())==0)
m = 0;
else
m = (float) (P->gety()-S->gety())/(P->getx()-S->getx());
float c = (float) (S->gety()) - (m*S->getx());
if(clip_edge==1){int x = xmax; int y = (m*x)+c;return (new points(x,y));} //bug was because of m=0 thing again
if(clip_edge==2){int y = ymax; int x; if(m==0) x = P->getx(); else x = (y-c)/m;return (new points(x,y));}
if(clip_edge==3){int x = xmin; int y = (m*x)+c;return (new points(x,y));}
if(clip_edge==4){int y = ymin; int x; if(m==0) x = P->getx(); else x = (y-c)/m;return (new points(x,y));}
}
float** out_to_in(float** inner, list<points*> out){
inner = new float*[out.size()];
for(int i=0; i<out.size(); i++){
inner[i] = new float[2];
}
```

```
sz = out.size();
while(out.size()>0){
points* temp = out.front();
inner[out.size()-1][0] = temp->getx();
inner[out.size()-1][1] = temp->gety();
out.pop_front();
}
out.empty();
return inner;
}
float** SHPC(float** inva, list<points*> out,int clip_edge){
/*cout<<"SHPC"<<endl;
for(int i=0; i<sz; i++)
cout<<"\n"<<inva[i][0]<<" "<<inva[i][1];
cout<<"\nxmin - "<<xmin<<" ymin - "<<ymin;
cout<<"\nxmax - "<<xmax<<" ymax - "<<ymax<<endl;*/
s = new points(inva[sz-1][0],inva[sz-1][1]);
for(int j=0; j<sz; j++){
p = new points(inva[j][0],inva[j][1]);
if(inside(p->getx(),p->gety(),clip_edge)) // case 1 & 4
{
if(inside(s->getx(),s->gety(),clip_edge)){ // case 1
out.push_front(new points(p->getx(),p->gety()));
}
else{ // case 4
points* temp = intersect(s,p,clip_edge);
out.push_front(temp);
out.push_front(p);
}
}
else if(inside(s->getx(),s->gety(),clip_edge)){ //case 2
```

```
points* temp = intersect(s,p,clip_edge);
out.push_front(temp);
}
else{
//cout<<"\nCASE3";
}
s = p;
}
inva = out_to_in(inva,out);
return inva;
}
void key(unsigned char key_t, int x, int y){
if((key_t=='d')||(key_t=='D'))
{
enter = -1;
glColor3f(0.0,0.0,1.0);
drawPolygon();
in.empty();
}
if((key_t=='c')||(key_t=='C'))
{
pts = SHPC(pts,outer,1);
pts = SHPC(pts,outer,2);
pts = SHPC(pts,outer,3);
pts = SHPC(pts,outer,4);
redraw();
}
}
void mouse(int btn, int state, int x, int y){
y = 480-y;
if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN && enter)
```

```
{
points* temp = new points(x,y);
in.push_front(temp);
draw_pixel(x,y);
}
}
void drag_start(GLint x, GLint y){
y = 480-y;
if(enter==-1&&st_flag){
xmin = x;
ymin = y;
st_flag = 0;
}
else{
xmax = x;
ymax = y;
}
redraw();
}
void drag_end(GLint x, GLint y){
y = 480-y;
if(enter==-1&&st_flag==0){
xmax = x;
ymax = y;
st_flag = 1;
redraw();
}
}
void world(){
glPointSize(2);
glClear(GL_COLOR_BUFFER_BIT);
```

```
glColor3f(1,0,0);
}
int main(int argc, char** argv){
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(640,480);
glutInitWindowPosition(200,200);
glutCreateWindow("Polygon Clipping");
glClear(GL_COLOR_BUFFER_BIT);
glClearColor(1.0,1.0,1.0,0);
glClear(GL_COLOR_BUFFER_BIT);
glutDisplayFunc(world);
glutMouseFunc(mouse);
glutMotionFunc(drag_start);
glutPassiveMotionFunc(drag_end);
glutKeyboardFunc(key);
init();
glutMainLoop();
return 0;
}
```